БИБЛИОСФЕРА, 2005, № 1, с. 47–54

Обзоры

УДК 025.4
ББК 78.30

ТЕХНОЛОГИИ ИНТЕГРАЦИИ ДАННЫХ, ПРИМЕНЯЕМЫЕ В СОВРЕМЕННОЙ ПРАКТИКЕ БИБЛИОТЕК

© В.Б. Федотов, 2005

Государственная публичная научно-техническая библиотека
Сибирского отделения Российской академии наук
630200, г. Новосибирск, ул. Восход, 15

Интеграция данных является одной из наиболее важных задач, решение которых должно обеспечиваться современными крупными информационными библиотечными системами. Все наиболее технологически совершенные решения этой задачи прошли долгий и тщательный эволюционный отбор. Основным и наиболее перспективным решением на сегодняшний день является виртуальная интеграция данных. Сегодня существует множество различных вариантов виртуальной интеграции данных, каждый из которых характеризуется своими достоинствами и недостатками.

Ключевые слова: интеграция; данные; распределённые информационные системы.

Наиболее острые проблемы, с которыми сталкивается сегодня информационное сообщество в целом и библиотеки как крупные информационные центры в частности, – это обеспечение надежного, бесперебойного, постоянного и полнофункционального доступа к актуальным данным.

Долгое время главенствующим подходом в этом вопросе была «**Физическая Интеграция Данных**», т.е. создание так называемых «**Хранилищ Данных**» (Data Warehouses) /1/. При этом все интегрируемые из разных источников данные трансформируются в соответствии с целевой моделью данных и хранятся централизованно на локальных, подключенных к серверу носителях, где регулярно обновляются и пополняются. Этому подходу посвящено много публикаций (см., например, /2/). Одним из способов хранения информации является Direct Attached Storage (DAS). Он кажется самым простым и «естественным». DAS действительно прост: если сервер включен, то всё внутреннее дисковое пространство доступно без ограничений /3/. Главными и несомненными достоинствами такого подхода является лёгкость администрирования собранных на одном сервере данных, полный контроль предоставляемых данных, унификация хранимых данных уже на физическом уровне. Однако на этом все преимущества заканчиваются. «Хранилища данных» будут жить и развиваться в дальнейшем, но их применение всё больше будет ограничиваться решением задач хранения и доступа к данным в рамках одной, достаточно унитарной организации. Особенности построения современных информационных хранилищ в сжатом виде приводятся А. Стуловым в его работе «Особенности построения информационных хранилищ» /4/. Более широкий обзор технологий, применяемых в этой области, можно найти у Э. Спирли /5/.

Популярность World Wide Web (WWW) превратила его в главное средство распространения информации /6/. На сегодняшний день бурное развитие Сети привело нас к тому, что единственным разумным способом справиться с нарастающим информационным потоком является организация сетевого способа обработки и хранения данных /3/. Поэтому следующим «эволюционным» шагом в развитии понятия «хранилища данных» стали системы класса NAS (Network Attached Storage) – сетевые системы хранения данных. Эти системы используют стандартные IP-сети и сетевые файловые системы. Именно NAS-системы пришли на смену DAS-структурам, подключаемым к сети, при организации общедоступных многотомных архивов. Подход NAS включает в себя файловую систему, используемую на базе соответствующей настраиваемой сетевой операционной системы, и файл-сервер, подключённый в сеть. В отличие от DAS, технология NAS поддерживает доступ на уровне файлов, а не блоков данных. Следствие этого отличия – «бесшовная» методика совместного использования файлов различными сетевыми

приложениями. Именно NAS берет на себя задачу трансляции обращения к конкретному файлу на запрос на уровне блоков данных и «экранирует» сетевые пользовательские процессы от проблем, связанных с обеспечением актуальности /3/.

Технология NAS более эффективна по сравнению с DAS, тем не менее в рамках сетевой концепции обработки данных она всё же не может быть признана «решением», хотя бы потому, что из нескольких NAS-устройств нельзя создать единый пул ресурсов хранения.

Можно сказать, что наиболее технологически развитым решением организации «хранилища данных» сегодня является технология SAN (storage area networking), предоставляющая для сетевой совокупности серверов консолидированный сетевой ресурс внешней памяти без нагрузки на локальную сеть. Консолидация в идеологии SAN строится на основе создания общего пула ресурсов, представляющих сетевую структуру с едиными принципами доступа. Вопросы организации SAN с точки зрения корпорации Sun Microsystems довольно подробно описываются В. Краюшкиным /3/. По версии Sun, консолидация ресурсов хранения должна быть реализована в виде виртуальной консолидированной памяти – Virtual Consolidated Store (VCS). Такая память реализуется прежде всего техническими средствами.

Идеология SAN подразумевает под собой физическое единообразие ресурсов. Кроме того, для реализации она требует решений на основе единых технических средств, что, несмотря на все её плюсы, делает её малоприменимой для доступа к распределённым по сети гетерогенным информационным ресурсам. Чаще всего проблема интеграции гетерогенных данных формулируется следующим образом: имеется несколько гетерогенных источников данных, которые каким-то образом связаны на смысловом уровне; необходимо предоставить возможность унифицированного доступа к этим данным, как если бы они имели единое логическое и физическое представление. Для решения этой проблемы потребовалась более гибкая альтернатива физической интеграции данных. Такой альтернативой стала «**Виртуальная Интеграция Данных**».

Виртуальная интеграция гетерогенных источников данных характеризуется тем, что данные не материализуются в локальной базе данных, вместо этого промежуточное программное обеспечение транслирует пользовательские запросы в подзапросы к источникам данных и на основе ответов отдельных источников данных формирует окончательный результат. Краткий обзор эволюции систем, использующих виртуальный подход, включая мультибазы данных /7/ и федеративные базы данных /8/, можно найти в зарубежных публикациях /9/. Подход этих систем характеризуется прежде всего тем, что интегрируются данные с четкой структурой (хотя структура данных разных источников может различаться). На следующем этапе появились системы интеграции, основанные на использовании медиаторов /10/ и создаваемые на базе полуструктурированных данных /11/. Возникновение XML /12/ и сопутствующих технологий (XSLT /13/, Xquery /14/) вызвало всплеск новых разработок технологии виртуальной интеграции (/15, 16/ и т.д.).

Развитие систем виртуальной интеграции данных вызвало резкое возрастание интереса к так называемым **распределенным системам**. Под распределенными системами обычно понимают программные комплексы, составные части которых функционируют на разных компьютерах в сети. Эти части взаимодействуют друг с другом, используя ту или иную технологию различного уровня – от непосредственного использования сокетов TCP/IP до технологий с высоким уровнем абстракции, таких, как RMI или CORBA /17/. Выбор приемлемой технологии создания распределённой информационной системы напрямую зависит от выбора архитектуры информационной системы (ИС).

Рассматривая информационную систему как совокупность взаимодействующих компонентов, можно распределить их по следующим уровням /18/:

I. Аппаратный уровень – компьютеры, периферийные устройства, сетевое и телекоммуникационное оборудование и т.д.

II. Системный и системно-зависимый уровни – операционные системы, сетевые протоколы и т.д.

III. Уровень прикладной среды – средства middleware (CORBA, DCE, Tuxedo и т.д.), DBMS, Intranet, OLAP, коммуникационные интерфейсы...

IV. Уровень приложения предметной области:

A. Общая инфраструктура – совокупность компонентов ИС, пригодных для использования в различных предметных областях. Такими компонентами, например, являются:

1. Средства автоматизации бизнес-процессов (атомарных задач и потоков работ).

2. Средства управления доступом к информационным ресурсам.

3. Средства составления и печати отчетов (генераторы отчетов).

B. Компоненты, реализующие модель предметной области.

Под проектированием архитектуры взаимодействия компонентов ИС (уровни II – IV) понимается выделение базовых компонентов, разработка их интерфейсов, а также определение правил и принципов взаимодействия этих компонентов. Каждый из таких компонентов представляет собой программный модуль, исполняемый в рамках отдельного процесса.

Проектирование архитектуры взаимодействия компонентов ИС – один из наиболее важных и сложных этапов, и ему не всегда уделяется достаточное внимание при разработке системы /18/.

При проектировании архитектуры взаимодействия распределенных компонентов информационной системы различают следующие типы взаимодействия /17/:

- **вертикальный** – каждый компонент имеет уникальный в рамках данной информационной системы интерфейс;
- **горизонтальный** – все компоненты имеют один и тот же универсальный интерфейс, обеспечивающий межкомпонентное взаимодействие;
- **смешанный** – все компоненты имеют универсальный базовый интерфейс, при этом каждый компонент специфицирует дополнительные операции для работы со своим доменом предметной области.

Достоинства и недостатки этих типов взаимодействия на примерах описываются К. Ахтырченко /18/.

Среди распределённых программных архитектур можно выделить следующие, наиболее распространенные классы:

- **двухуровневые архитектуры,**
- **трехуровневые архитектуры,**
- **распределенные одноранговые архитектуры.**

Двухуровневая архитектура предполагает, что количество уровней равно двум. Это означает наличие максимум двух программных компонентов, в совокупности реализующих все функции из следующих трех основных групп:

1) группа функций пользовательского интерфейса;

2) группа прикладных функций, характерных для данной предметной области;

3) группа функций хранения и управления данными.

Один из двух компонентов выступает в роли сервера, т.е. реализует набор сервисов, доступных другому компоненту, который выступает в роли клиента, т.е. в процессе работы пользуется сервисами, предоставляемыми сервером. Компоненты могут располагаться как на одном компьютере, так и на нескольких компьютерах, объединенных в сеть.

Трехуровневая архитектура предполагает, что количество уровней равно трём. Это означает наличие максимум трех программных компонентов, которые участвуют в процессе, предполагающем выполнение функций из первой, второй и третьей групп.

Трехуровневые архитектуры предусматривают не столь жесткие связи между клиентом и сервером и более гибкие формы распределенной обработки /19/. Отдельные компоненты могут располагаться как на одном компьютере, так и на разных компьютерах, обеспечивая тем самым распределенную обработку информации.

Согласно **распределенной одноранговой архитектуре** /19, 20/ клиент, взаимодействующий с сервером приложений, трактуется более широко. Он может поддерживать интерфейс с конечным пользователем, а может выполнять прикладные функции и являться сервером приложения. В общем случае в рамках данной архитектуры клиент (сервер) может как предоставлять, так и запрашивать некоторые сервисы. Это позволяет на этапе проектирования информационной системы осуществить такую декомпозицию функций из указанных выше трех групп по компонентам ИС, которая была бы оптимальной в контексте решаемой задачи /18/.

Ключевым элементом, определяющим построение любой распределённой информационной системы, выступают **технологии интеграции компонентов ИС.** Для оценки существующих технологий интеграции компонентов информационных систем можно выделить следующие *категории* технологических решений, характеризующие воззрение конкретной организации на архитектуру информационной системы в целом /21, 22/:

1. Частные решения.

2. Разнообразные (смешанные) механизмы (Miscellaneous Mechanisms).

3. Удаленные вызовы процедур на базе DCE RPC.

4. Распределенные объекты (CORBA, DCOM) (Distributed Object).

5. Frameworks.

6. Стандартные архитектуры (Standard Architectures).

Технологические решения, относящиеся к *первой категории*, базируются на основе уникальных для данной организации протоколов и интерфейсов взаимодействия. Такие решения часто порождают непреодолимые трудности при попытке организации общения компонентов

данной ИС с компонентами ИС, построенных на основе других интеграционных решений.

Ко *второй категории* относятся технологические решения, предполагающие построение информационных систем из расчета на конкретную задачу и изначально не рассчитанные на использование технологий интеграции систем. Поэтому в данном случае в качестве средства интеграции, как правило, используется ONC RPC (Object Network Computing Remote Procedure Call) или механизмы, обеспечивающие работу с сетевыми протоколами, например сокеты (sockets) для работы с протоколом TCP/IP. Подробнее о сетевых протоколах и TCP/IP можно прочитать у Jeremy Bentham /23/.

В технологических решениях *третьей категории* предполагается использование технологий, базирующихся на средствах единообразной интеграции компонентов. К данной категории относятся технологии на базе механизма вызова удаленных процедур (RPC), такие, как OSF DCE. Как известно, OSF DCE определяет сервисы безопасности, именования и другие важные механизмы, необходимые для интеграции систем в распределенной среде. Характерным недостатком OSF DCE считается сложность создания объектов как независимых компонентов распределенной системы и ориентированность на процедурный стиль программирования. Поэтому имеется тенденция рассматривать DCE как технологию, которая устарела в сравнении с новыми объектно-ориентированными технологиями построения распределенных систем /17/.

В *четвертой категории* при построении информационных систем используются ORB-технологии CORBA, такие, как высокоуровневый механизм RPC. Здесь применяют сервисы и язык описания интерфейсов (OMG IDL), определенные в спецификации CORBA, только для обеспечения межплатформенного взаимодействия. В случае, когда межплатформенное взаимодействие не требуется, используются другие механизмы взаимодействия компонентов системы. Например, технологические решения второй категории – сокеты (sockets) протокола TCP/IP или ONC RPC.

Технологические решения *пятой категории* предполагают разработку frameworks как основы программной архитектуры для использования в нескольких проектах. Как правило, framework воплощает тщательно продуманные принципы построения программной архитектуры. Разработка frameworks – направление многообещающее и бурно развивающееся в последнее время, более подробно про создание frameworks можно прочитать в обзоре /24/.

Наконец, к *шестой категории* относятся высококачественные технологии, программные архитектуры и сервисы, которые рассчитаны на повторное использование во множестве различных проектов. Не каждая организация способна создать технологию мирового класса, поэтому организации, разрабатывающие решения шестой категории, – это всемирно известные фирмы, являющиеся создателями стандартов взаимодействия систем в своей отрасли /18/.

Можно выделить следующие наиболее популярные решения, относящиеся к шестой категории.

Технологии взаимодействия объектов

COM-технология создавалась фирмой Microsoft как средство взаимодействия приложений (в том числе составных частей операционной системы) Windows, функционирующих на одном компьютере, с последующим развитием для использования в пределах локальной сети. Главная задача на момент создания – обеспечение технологии Object Linking and Embedding (OLE 1.0). Характерно, что обмен данными между приложениями (Dynamic Data Exchange, DDE) первоначально строился не по COM-технологии, а с использованием достаточно популярного механизма сообщений (messages), эволюция которого привела к появлению ряда самостоятельных технологий интеграции компонент. Концепция технологии COM неразрывно связана с ее реализацией. Технология характеризуется двоичной структурой объекта, хотя в настоящий момент уже существует и язык описания структуры объекта – Interface definition Language (IDL).

CORBA-технология создавалась консорциумом OMG как универсальная технология создания распределенных систем в гетерогенных средах. В настоящий момент в OMG состоит более 800 членов, включая всех сколько-нибудь серьезных производителей программного обеспечения (и даже с недавнего времени Microsoft). Первая спецификация CORBA появилась в 1991 г. Новые возможности официально считаются добавленными в CORBA в момент утверждения соответствующей спецификации. Разработка реализации – задача конкретной фирмы. Обычно от утверждения спецификации до появления высококачественной реализации проходит довольно много времени – иногда несколько лет. Характерным элементом технологии являются объявления на языке IDL, который является «сердцем» CORBA с момента ее появления.

Несмотря на внешнюю похожесть, что вызвано общностью решаемых задач, между COM

и CORBA, пожалуй, больше различий, чем сходства. В большинстве случаев либо нецелесообразно использовать CORBA (для небольших и простых проектов под Windows просто по причине относительно высоких затрат на приобретение программного обеспечения, лицензий и пр.), либо практически невозможно использовать COM (для сложных, масштабируемых, высоконадежных проектов или просто при работе в гетерогенных средах, а не только в Windows). Windows-приложения, ориентированные на взаимодействие с Microsoft Office, всегда будут использовать COM; проекты с использованием Java значительно удобнее строить на основе CORBA. Сравнительный анализ технологий CORBA и COM приводит А. Цимбал /17/.

Технологии систем очередей сообщений (Messaging Oriented Middleware – MOM)

В Windows существуют два средства, с помощью которых можно реализовать асинхронное взаимодействие в распределенных информационных системах – это **MS Message Queue Server (MSMQ Server) и Queued Components (QC)**.

MSMQ включает в себя большой набор возможностей, позволяющих приложениям, расположенным на различных машинах, посылать и принимать асинхронные сообщения, тем самым обеспечивая базовый сервис доставки сообщений (basic message delivery service). Приложение, которое должно принимать сообщения, создает очередь – сохраняемую административную структуру, доступную операционной системе и являющуюся, по сути дела, почтовым ящиком. Приложение, которое должно посылать сообщения, размещает эту очередь через системные сервисы или собственными силами и использует MSMQ для посылки сообщений. Если компьютер получателя с очередью недоступен на момент посылки сообщения, сообщение буферизируется операционной системой передающей машины либо другими сетевыми машинами до момента доступности получателя. Приложение получателя может принимать сообщения MSMQ путем опроса очереди сообщений или по уведомлениям обратного вызова (callback notifications). MSMQ гарантирует, что все передаваемые сообщения в конечном счете достигнут адресата. Кроме того, эта технология обеспечивает эффективную маршрутизацию, защиту и нужную приоритетность. MSMQ также обеспечивает функциональность, необходимую для управления посылаемыми сообщениями. Небольшой обзор MSMQ можно найти у Александра Эпштейна /25/, подробное же описание возможностей MSMQ можно прочитать в книге Девида Чапела /26/. По функциональным возможностям и архитектуре MSMQ в значительной степени похожа на разработку IBM – MQSeries /27/.

QC-компоненты являются модификацией COM+/MTS-компонентов и обеспечивают более высокий уровень абстракции, чем MSMQ. Queued Components отвечают требованиям механизма Store-and-Forward. Клиентское приложение может использовать QC-компоненты так же, как любой другой COM-объект. Они основаны на COM (Component Object Modeling), но в качестве транспортного протокола вместо RPC используют MSMQ-сервис. Во время активизации QC-объекта клиент подключается не к COM-объекту, а к Call Recorder. Клиентское приложение производит вызовы как обычно, но они не отправляются немедленно через RPC. Вместо этого информация о них записывается в очередь MSMQ. Когда клиент деактивирует компонент, QC использует MSMQ для асинхронной посылки последовательности вызовов серверу, на котором создается реальный компонент. Сервер читает сообщения из MSMQ-очереди, содержащей эту последовательность, активизирует компонент Player (исполнитель) и передает ему информацию о вызовах. Player создает реальный объект и воспроизводит на нем вызовы.

Детальное сравнение возможностей технологий MSMQ и QC проводится А. Новиком /28/.

История **семейства MQSeries** как единого семейства программных продуктов начинается с 1992 г., когда компания IBM опубликовала спецификации для программного интерфейса Message Queue Interface (MQI).

Сегодня менеджеры очередей MQSeries работают на OS/390, MVS, VSE/ESA, OS/400, OS/2, Tandem Guardian и Himalaya, OpenVMS VAX, Digital Unix, AIX, HP-UX, SunOS, Sun Solaris, SCO UNIX, UnixWare, AT&T GIS UNIX, DC/OSx, Windows NT/95/3.1. Для еще большего числа платформ, в том числе для DOS, Java, MacOS, Linux, существуют MQSeries клиенты. Взаимодействие менеджеров очередей MQSeries даже разных версий происходит прозрачно для внешних программ, что обеспечивает им единый интерфейс MQI и функционирование единой транспортной системы MQSeries. Можно указать ряд направлений развития MQSeries и всей архитектуры средств MOM: появляются новые прикладные и административные интерфейсы, упрощающие процесс создания новых систем; поддерживаются более сложные модели обработки сообщений, такие как публикация-подписка или обработка с анализом контекста сообщений; развивается интеграция между

MQSeries и реляционными базами данных; появляются решения для поддержки совместной работы нескольких менеджеров очередей, соединенных в кластеры.

Основные элементы архитектуры системы управления очередями сообщений IBM MQSeries описываются Н. Игнатовичем /27/.

Использование очередей сообщений существенно отличается от RPC (удаленного вызова процедур), Windows Sockets и MAPI (messaging API – e-mail-ориентированный сервис) тем, что реализует не ориентированный на подключение способ передачи сообщений, когда взаимодействующие приложения не обязаны работать одновременно и/или ориентироваться на соединение. RPC-приложения ориентированы на соединение, Windows Sockets работают только с одновременно работающими приложениями. А MAPI, хотя и отвечает основным требованиям систем с промежуточным накоплением, уступает MSMQ в универсальности и ориентируется только на электронную почту. Многие аналитики компьютерной индустрии, например специалисты из Gartner Group, отмечают сегодня быстрый рост числа решений, использующих очереди сообщений, и подчёркивают при этом гибкость и адаптируемость подобной архитектуры. Системы очередей сообщений предоставляют услуги сохранения сообщений и последующей их доставки другой программе (метод store and forward). Прикладная программа передает свое сообщение серверу (менеджеру очередей), который записывает его в локальную очередь, а затем передает по сети другому менеджеру очередей, содержащему очередь-адресат. Программа-адресат обращается к целевой очереди и получает доступ к сообщению. В результате система очередей сообщений предоставляет асинхронный метод взаимодействия программ, не требующий установки между ними прямой связи. При этом гарантируется, что передаваемое сообщение не будет потеряно или получено дважды /28/.

Использование программного обеспечения, предоставляемого системами очередей сообщений в качестве асинхронного механизма для организации взаимодействия между программами, позволяет создать унифицированную транспортную шину для транзакционных и информационных вычислительных систем различного масштаба, обеспечивая гибкое решение для организации асинхронного взаимодействия между программами в распределенной среде.

Семантическая интеграция

Другим активно развивающимся направлением интеграции является семантическая интеграция. К этой группе технологий можно отнести такие активно применяющиеся в библиотечном и не только в библиотечном деле технологии, как Z39.50, XML, RDF, SOAP.

В основе **Z39.50** лежит идея построения абстрактной модели работы с абстрактной базой данных. Каждый элемент этой абстрактной модели подробно описывается до однозначного толкования и стандартизуется с присвоением уникального идентификатора – OID. Работа с каждой конкретной системой управления базами данных (СУБД) согласно Z39.50 должна быть организована только через эту абстрактную модель, что позволяет, с одной стороны, однозначно отобразить логику запроса, абстрагируясь от синтаксиса запроса конкретной СУБД, а с другой – абстрагироваться от поисковых полей конкретной базы данных. Структурированные форматы внешнего представления позволяют после передачи по сети полностью сохранить первоначальную структуру записи, что является немаловажным в распределенных системах /29/.

Подробное описание протокола Z39.50 приводится О.Л. Жижимовым /30/.

Язык **XML** предоставляет нам чрезвычайно удобный и почти универсальный подход к хранению и передаче информации. Обмен информацией в формате XML – это механизм, позволяющий свести к минимуму проблемы внутрифирменных форматов данных. Эти проблемы сводятся к сложностям при обмене информацией с контрагентами фирмы. Зачастую проблема состоит не столько в невозможности других разработчиков информационных систем принимать предлагаемый частный формат данных (DBF, ASCII и т.п.), сколько в нежелании приспосабливаться к нему /31/.

Язык XML позволяет разрабатывать форматы описания данных, которые могут быть использованы как мостики, которые связывают отдельные частные источники данных и устраняют технические и психологические барьеры, неизбежно возникающие при приспособлении к чужим технологиям. При этом формирование документов в формате XML является достаточно простым процессом, для этого нужно лишь ознакомиться с конкретным описанием структуры данных (DTD).

Подробнее об языке XML излагается в соответствующих рекомендациях консорциума W3C /12/.

RDF. При крайнем разнообразии семантики описываемых ресурсов невозможно создать единое семантическое пространство, которое удовлетворяло бы все потребности в описании семантики ресурсов. В то же время каждый ресурс,

а точнее, группа связанных ресурсов, как правило, имеет ограниченную семантику. Описывать семантику ресурса и затем выполнять поиск тем проще, чем уже и специфичнее его предметная область. При явной невозможности «слить» воедино все значимые понятия, встречающиеся в ресурсах, остается только смириться с вынужденной неоднородностью и независимостью их семантических описаний.

В применении к распределенному поиску это значит, что все семантические элементы искомого ресурса должны быть известны, по крайней мере, той поисковой системе, которая этот ресурс непосредственно обслуживает, а все посредники в цепочке передачи запроса должны корректно «пронести» семантику от ресурса до запрашиваемого посредника путем передачи ее в виде отражений («снизу вверх»), чтобы запрос нашел путь от запрашиваемой системы до оконечной («сверху вниз») /32/.

Для описания предметной области ресурсов (под ресурсом можно понимать источники масштаба от документа до группы сайтов) может быть использован стандарт **RDF** /33/, принятый в 1999 г. консорциумом W3C и поддержанный многими ведущими производителями программного обеспечения и поставщиками контента.

SOAP – это новый «лёгкий» протокол, предназначенный для обмена структурированной информацией в децентрализованной, распределённой среде. Он использует XML-технологии для создания масштабируемых структур обмена сообщениями, предоставляя конструирование сообщений, которыми можно обмениваться при помощи множества различных протоколов. Структура SOAP была разработана независимой от любой конкретной программной модели или конкретной реализации специфической семантики. Двумя основными задачами разработки SOAP были простота и расширяемость. SOAP пытается достичь этих целей путём создания структур обмена сообщениями, обладающих свойствами, которые часто могут быть найдены в распределённых системах. Такие свойства включают в себя, но не ограничиваются следующими аспектами: «надёжность», «защищённость», «соотношение», «формальность» и «модели обмена сообщениями» «Message Exchange Patterns» (MEPs). Ключевым в спецификации SOAP являются модели обмена сообщениями, остальные же свойства пока определяются как расширения будущих спецификаций.

Детальную информацию о протоколе SOAP можно найти в соответствующих рекомендациях консорциума W3C /34, 35/.

Выводы

Важен тот факт, что для всех технологий семантической интеграции характерно функционирование поверх какого-то транспортного уровня, что даёт большие просторы для возможной комбинации семантических технологий с технологиями взаимодействия распределённых компонентов информационной системы, для достижения максимально эффективных по производительности, масштабируемости и универсальности решений виртуальной интеграции гетерогенных информационных ресурсов.

Список источников

1. *Гринев М.* BizQuery: система виртуальной интеграции данных на основе языка XML [Электронный ресурс] // ИСП РАН, фак. ВМиК МГУ. – Режим доступа: http://www.citforum.ru/seminars/cbd 2003/2_05_grinev/2_05_grinev.shtml
2. A selection of papers on datawarehousing // Computer. – 2001. – Vol. 14, N 12.
3. *Краюшкин В.* Виртуальная консолидация данных [Электронный ресурс]. – Режим доступа: http:// www.osp.ru/os/2003/04/008_print.htm
4. *Стулов А.* Особенности построения информационных хранилищ [Электронный ресурс]. – Режим доступа: http://www.osp.ru/os/2003/04/076_print.htm
5. *Спирли Э.* Корпоративные хранилища данных. Планирование, разработка, реализация. – М.: Вильямс, 2001. – Т. 1.
6. *Флореску Д., Леви А., Мендельсон А.* Технологии баз данных для World Wide Web: Обзор [Электронный ресурс]. – Режим доступа: http://www.osp. ru/dbms/1998/04-05/01_print.htm
7. *Batini C., Lenzerini M., Navathe S.* A Comparative Analysis of Methodologies for Database Schema Integration // ACM Computer Surveys. – 1986. – Vol. 18(4). – P. 323–364.
8. *Sheth A., Larson J.* Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases // ACM Computer Surveys. – 1986. – Vol. 22(3). – P. 183–236.
9. *Grinev M., Kuznetsov S.* UQL: A Query Language on Integrated Data in Terms of UML // Programming and Computer Software. – 2002. – Vol. 28, N 4. – P. 189–196.
10. *Wiederhold G.* Mediators in the Architecture of Future Information Systems // IEEE Computer. – 1992. – Vol. 25(3). – P. 38–49.
11. The TSIMMIS Project: Integration of Heterogeneous Information Sources / Chawathe S., Garcia-Molina H., Hammer J. et al. // IPSJ. – 1994. – P. 7–18.
12. Extensible Markup Language (XML) 1.0, W3C Recommendation, 2nd edition [Электронный ресурс]. – Режим доступа: http://www.w3.org/TR/2000/REC-xml-20001006
13. XSL Transformations (XSLT) 2.0, W3C Working Draft 15 November 2002 [Электронный ресурс]. –

Режим доступа: http://www.w3.org/TR/2002/WD-xslt 20-20021115/
14. XQuery 1.0: An XML Query Language, W3C Working Draft 15 November 2002 [Электронный ресурс]. – Режим доступа: http://www.w3.org/TR/xquery/
15. The tukwila data integration system, university of Washington [Электронный ресурс]. – Режим доступа: http://data.cs.washington.edu/integration/tukwila/
16. Xperanto project, IBM almaden research center [Электронный ресурс]. – Режим доступа: http:// www.almaden.ibm.com/software/dm/Xperanto/index. shtml
17. *Цимбал А.* Сравнительный анализ технологий CORBA и COM [Электронный ресурс]. – Режим доступа: http://www.interface.ru/fset.asp?Url=/borland /corbacom.htm
18. *Ахтырченко К.В., Леонтьев В.В.* Распределенные объектные технологии в информационных системах [Электронный ресурс]. – Режим доступа: http://osp.admin.tomsk.ru/ dbms/1997/05_06/52.htm
19. *Эккерсон В.* В поисках лучшей архитектуры клиент-сервер // Сети. – 1995. – № 4.
20. Guide to building client/server solutions, Digital Equipment Corporation. – San Paolo, California, 1993.
21. *Mowbray T.J., Zahavi P.R.* The Essential CORBA: System Integration Using Distributed Object. – 1995.
22. *Orfali R., Harkey D., Edwards J.* The Essential Distributed Object. – New York: John Wiley&Sons, Inc., 1996.
23. *Jeremy B.* TCP/IP Lean Web Servers for Embedded Systems, Second Edition. – CMP Books, CMP Media LLC, 2002.
24. Построение объектно-ориентированных сред разработки (Frameworks) [Электронный ресурс]. – Режим доступа: http://www.devresource.org/java/17. htm
25. *Эпштейн А.* Асинхронная передача сообщений с помощью MSMQ [Электронный ресурс]. – Режим доступа: http://www.osp.ru/win2000/2001/08/050_print. htm
26. *Chappell D.* Microsoft Message Queue is a Fast, Efficient Choice for Your Distributed Application. – MSJ, 1998.
27. *Игнатович Н.* IBM MQSeries: архитектура системы очередей сообщений [Электронный ресурс] // Открытые системы. – 1999. – № 9/10. – Режим доступа: http://www.citforum.elcat.kg/programming/digest /ibmmqs.shtml
28. *Новик А.* Queued-компоненты Windows 2000 [Электронный ресурс] // Технология клиент-сервер. – 1999. – № 3. – Режим доступа: http://www.optim.ru/cs/1999/3/qc/qc.asp
29. *Баженов С.Р., Баженов И.С., Федотов В.Б.* Совершенствование Web-ориентированной системы управления базами данных CDS/ISIS // Библиотеки и ассоциации в меняющемся мире: новые технологии и новые формы сотрудничества: Материалы междунар. конф. «Крым-2002» (Судак). – М., 2002. – Т. 1. – С. 172–175.
30. *Жижимов О.Л.* Введение в Z39.50. – Новосибирск: Изд-во НГОНБ, 2000. – 196 с.
31. *Чудин А.* Загрузка и анализ документа XML [Электронный ресурс]. – Режим доступа: http://delphi. mastak.ru/articles/Load-XML/index.htm
32. *Жигалов В.А.* Интернет как распределенная поисковая система [Электронный ресурс] // Научный сервис в сети Интернет: Тез. докл. Всерос. науч. конф. – Режим доступа: http://www.aha.ru/~zhigalov/science/articles/InetAsDSS.htm
33. Resource Description Framework (RDF) Model and Syntax Specification W3C Recommendation 22 February 1999 [Электронный ресурс]. – Режим доступа: http://www.w3.org/TR/REC-rdf-syntax/
34. SOAP Version 1.2 Part 1: Messaging Framework W3C Recommendation 24 June 2003 [Электронный ресурс]. – Режим доступа: http://www.w3.org/TR/ 2003/REC-soap12-part1-20030624/
35. SOAP Version 1.2 Part 2: Adjuncts W3C Recommendation 24 June 2003 [Электронный ресурс]. – Режим доступа: http://www.w3.org/TR/2003/REC-soap 12-part2-20030624/

Материал поступил в редакцию 19.11.2004 г.

Сведения об авторе: *Федотов Вадим Борисович – аспирант ГПНТБ СО РАН, инженер отдела автоматизированных систем, тел. (383-2) 66-75-79, e-mail: fedot_st@online.nsk.su*